DETAILED ACTION

Status of Claims:

Claims 1-9, 11-21, 23-25, and 28-32 are pending in this Office Action.

Claims 1-9, 11-21, 23-25 and 28 are amended.

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 1-9, 11-21, 23-25, and 28-32 are rejected under 35 U.S.C. 103(a) as being unpatentable over Davis (US 2003/0154239) in view of Jeyaraman (US 2003/0236923).

Claim 1

Davis teaches a system comprising:

a plurality of instances of an application server coupled in a star topology with the message server at a center of the star topology, the message server handling communications between the plurality of instances of the application server **(paragraph [0014] "According to another aspect of the present invention, developers preferably separate their Web application into two layers: a highly distributed edge layer and a centralized origin layer. In a representative embodiment, the**

edge layer supports a Web container so that the following technologies are supported: Java server pages (JSPs), servlets, Java beans, Java helper classes, and tag libraries. Preferably, communications between the edge and the origin use conventional communication protocols such as RMI and SOAP. Any protocol that can be tunneled over HTTP, such as JDBC, can also be supported" and paragraph [0015] "Preferably, an application is run on the edge server in its own application server instance in its own Java virtual machine (JVM). In a preferred embodiment, a content delivery network service provider operates a CDN with at least one edge server that includes multiple application server/JVM instances, with each instance associated with a given CDN customer"), one or more of the plurality of instances to register or reregister instance-specific information with the message server upon a starting or restarting, respectively, of the message server **(paragraph [0060] "In particular, the wrapper 1006 initializes JESAPI 1004, performs any necessary runtime configuration of the application server process 1002, starts the server, and notifies JESAPI when the server is ready to process requests. Because it is the entry point for the application, the wrapper must initialize JESAPI and the application server with the data supplied to it by the edge server process (element 900 in FIG. 9) (in the form of arguments, Java system properties, and the like). The data includes, for example: an application server instance id (used by JESAPI) and the socket port the servlet container must be on for HTTP connections")**.

Davis does not disclose a message server that is configured to be restarted after a failure without performing state recovery operations; and the instance-specific information identifying one or more services that the one or more of the plurality of instances is configured to provide to each of the plurality of instances.

Jeyaraman teaches in paragraph [0033] “Once message endpoints are activated, they are ready to receive messages from a message provider. Message delivery setup could either be durable or non-durable. In the case of non-durable message deliveries, messages are lost during application server downtime. When the application server becomes functional again, it automatically reactivates all message endpoints that were previously deployed, and messages delivery starts again. But the messages that were produced during the downtime are lost. This is because messages are not retained by the message provider and redelivered when the message endpoints are reactivated” and in paragraph [0005] “In modern Enterprise Information Systems (EIS) is formed to include a number of EIS resources. An EIS resource provides EIS-specific functionality to its clients examples of which include a record or set of records in a database system, a business object in an ERP system, and a transaction program in a transaction processing system” and paragraph [0007] “The J2EE Connector architecture defines standard contracts which allows bi-directional connectivity between enterprise applications and EISs. An architecture for integration of J2EE servers with EISs is referred to as a connector architecture. There are two parts to the connector architecture: an EIS vendor-provided resource adapter and an application server that allows this resource adapter to plug in. The contracts support bi-directional

communication between the application server and the resource adapter" in order that "Once endpoints are activated, they are ready to receive messages. When messages arrive, the resource adapter uses the message endpoint factory to create an endpoint instance. The resource adapter narrows the endpoint instance to the actual message listener type (it knows the endpoint type from the Activation Spec), and delivers the message to the endpoint instance" (paragraph [0032]).

It would have been obvious to one of ordinary skill in the art at the time to create the invention of Davis to include the non-durable message server and the identifying services as taught by Jeyaraman in order that "Once endpoints are activated, they are ready to receive messages. When messages arrive, the resource adapter uses the message endpoint factory to create an endpoint instance. The resource adapter narrows the endpoint instance to the actual message listener type (it knows the endpoint type from the Activation Spec), and delivers the message to the endpoint instance" (paragraph [0032]).

Claims 14 and 16 are rejected for the same reasoning as claim 1 as they are analogous in scope.

Claim 2

The modified Davis teaches the system of claim 1 wherein each of the plurality of instances comprises:

a dispatcher node; and a plurality of server nodes **(paragraph [0058] "The following describes modifications to a Java application server, specifically its servlet container component, to integrate into the inventive framework. This application server is executed on an edge server, which, as noted above, is a machine running commodity hardware and an operating system. As illustrated in FIG. 9, a preferred architecture is implemented via out of process architecture and comprises an edge server process 900 and multiple Java application server processes 902a-n. An edge node in the content delivery network preferably has a single edge server application that can spawn multiple child processes each containing an application server instance, as was illustrated in FIG. 8. Each child process preferably is configured for a Java Edge Services API (JESAPI), which according to the invention is an integration framework for a Java application server")**.

Claims 8 and 20 are rejected for the same reasoning as claim 2 as they are analogous in scope.

Claim 3

The modified Davis teaches the system of claim 2 wherein each of the plurality of server nodes comprises:

a java 2 enterprise edition (J2EE) engine **(paragraph [0042] "The present invention is a CDN Java application framework offering comprising Java-enabled**

edge servers. A given edge server (the machine) such as illustrated above in FIG. 2 is assumed to include application server code. As is well-known, an application server is a software platform (sometimes called middleware) on which applications can be deployed. It provides useful utility services and functions to applications. There are currently several major types of application servers, Java-based (J2EE) and Microsoft .NET. Java, of course, is a programming language and a platform, and the programming language is object-oriented and platform independent").

Claim 4

The modified Davis teaches the system of claim 1 further comprising:

a central lock server to provide cluster wide locks to the plurality of instances **(paragraph [0015] "In addition to resource management, preferably security restrictions are imposed on applications running in each application server/JVM process. This is sometimes referred to as sandboxing. These restrictions include, for example, file system read/write restrictions, limitations on socket opening and usage, restrictions on thread starting, stopping and modification, as well as code restrictions that prevent applications from reading certain application server classes. Preferably, a given application cannot run or load code belonging to other applications, it cannot load data belonging to another application, it cannot read or write arbitrary files on the file system, and it cannot make native kernel calls or load libraries that make native calls")**.

Claims 6, 9, 11, 12, 15, 18, 21, 23 and 24 are rejected for the same reasoning as claim 4 as they are analogous in scope.

Claim 5

The modified Davis teaches the system of claim 1 wherein the message server comprises:

a first data structure to store a list of connected clients **(paragraph [0048] "At step (2), the edge server 602 applies the customer's configuration data 610 to the request, determining if the request should be serviced using the edge server's local cache 608 or Java processor 606, or forwarded (e.g., via tunneling) to the customer's origin server 604. Thus, when the edge server receives a request from a client, preferably it first matches the request with an appropriate customer configuration file. If the customer configuration file associates Java processing with the request, the Java processor 606 is engaged. If the request is for a servlet or a JSP page, the Java processor 606 fulfills the request")**; and a second data structure to store a list of services provided in the system **(paragraph [0048] "FIG. 6 illustrates how an end user client browser 600 interacts with a content delivery network edge server 602 and an origin site 604 to facilitate execution of the application (and, in particular, its Web tier components) on the edge of the network. In this example, it is assumed that the Web tier components of the application are available for deployment and execution on the edge server. As**

described above, the edge server 602 has a Java processor 606, a cache 608, and a set of customer configuration data 610. The origin site 604 executes a Java application server 612 and includes data sources 614").

Claim 17 is rejected for the same reasoning as claim 5 as it is analogous in scope.

Claim 7

The modified Davis teaches the non-transitory computer readable storage media of claim 6, the method further comprising sharing a database among the plurality of application server instances **(paragraph [0013] "The Enterprise tier typically comprises middleware such as entity beans, session beans, and message-driven beans that implement the application's business logic and that provide local or remote database support")**.

Claim 19 is rejected for the same reasoning as claim 7 as it is analogous in scope.

Claim 13

The modified Davis teaches the non-transitory computer readable storage media of claim 10, the method further comprising notifying all registered instances from the message server when an additional instance joins the cluster **(paragraph [0060] "The**

application wrapper 1006 acts as the bootstrap logic for the application server process 1002. The wrapper 1006 is customized to the application server type and acts as "glue" code connecting all the various components of the process. The wrapper component 1006 provides a JESAPI implementation singleton specific for the application server type, which may vary. In particular, the wrapper 1006 initializes JESAPI 1004, performs any necessary runtime configuration of the application server process 1002, starts the server, and notifies JESAPI when the server is ready to process requests. Because it is the entry point for the application, the wrapper must initialize JESAPI and the application server with the data supplied to it by the edge server process (element 900 in FIG. 9) (in the form of arguments, Java system properties, and the like). The data includes, for example: an application server instance id (used by JESAPI) and the socket port the servlet container must be on for HTTP connections").

Claims 25 and 28 are rejected for the same reasoning as claim 13 as they are analogous in scope.

Claim 29

The modified Davis teaches the system of claim 1, wherein each of the plurality of instances is started using a first instance-specific bootstrap logic, the first instance-specific bootstrap logic synchronized with a second instance-specific bootstrap logic stored in the database **(paragraph [0060] "The application wrapper 1006 acts as the**

bootstrap logic for the application server process 1002. The wrapper 1006 is customized to the application server type and acts as "glue" code connecting all the various components of the process. The wrapper component 1006 provides a JESAPI implementation singleton specific for the application server type, which may vary. In particular, the wrapper 1006 initializes JESAPI 1004, performs any necessary runtime configuration of the application server process 1002, starts the server, and notifies JESAPI when the server is ready to process requests. Because it is the entry point for the application, the wrapper must initialize JESAPI and the application server with the data supplied to it by the edge server process (element 900 in FIG. 9) (in the form of arguments, Java system properties, and the like). The data includes, for example: an application server instance id (used by JESAPI) and the socket port the servlet container must be on for HTTP connections. The application wrapper 1006 preferably configures the edge server to only accept HTTP socket connections. In an illustrative embodiment, the application server process must accept connections bound for the local loopback host and on the port specified by the edge server process. Additionally, the application wrapper provides and registers any handlers with the application server necessary for integration, such as protocol handling and logging").

Claim 30

The modified Davis teaches the system of claim 1, wherein a node within the plurality of instances is started using a first node-specific bootstrap logic, the first node-specific bootstrap logic synchronized with a second node-specific bootstrap logic stored in the database **(paragraph [0060] "The application wrapper 1006 acts as the bootstrap logic for the application server process 1002. The wrapper 1006 is customized to the application server type and acts as "glue" code connecting all the various components of the process. The wrapper component 1006 provides a JESAPI implementation singleton specific for the application server type, which may vary. In particular, the wrapper 1006 initializes JESAPI 1004, performs any necessary runtime configuration of the application server process 1002, starts the server, and notifies JESAPI when the server is ready to process requests. Because it is the entry point for the application, the wrapper must initialize JESAPI and the application server with the data supplied to it by the edge server process (element 900 in FIG. 9) (in the form of arguments, Java system properties, and the like). The data includes, for example: an application server instance id (used by JESAPI) and the socket port the servlet container must be on for HTTP connections. The application wrapper 1006 preferably configures the edge server to only accept HTTP socket connections. In an illustrative embodiment, the application server process must accept connections bound for the local loopback host and on the port specified by the edge server process. Additionally, the application wrapper provides and registers any handlers with**

the application server necessary for integration, such as protocol handling and logging").

Claim 31

The modified Davis teaches the method of claim 18, wherein the instance-specific information further includes information about a new service that the one or more of the plurality of instances provide **(paragraph [0060] "The application wrapper 1006 acts as the bootstrap logic for the application server process 1002. The wrapper 1006 is customized to the application server type and acts as "glue" code connecting all the various components of the process. The wrapper component 1006 provides a JESAPI implementation singleton specific for the application server type, which may vary. In particular, the wrapper 1006 initializes JESAPI 1004, performs any necessary runtime configuration of the application server process 1002, starts the server, and notifies JESAPI when the server is ready to process requests. Because it is the entry point for the application, the wrapper must initialize JESAPI and the application server with the data supplied to it by the edge server process (element 900 in FIG. 9) (in the form of arguments, Java system properties, and the like). The data includes, for example: an application server instance id (used by JESAPI) and the socket port the servlet container must be on for HTTP connections")**.

Claim 32

The modified Davis teaches the system of claim 1, wherein the plurality of instances are unable to communicate with each other during a failure of the message server **(See claim 1 rejection; restarting of message server)**.

Response to Arguments

3. Applicant's arguments with respect to claims have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

4. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of

the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to FARHAD ALI whose telephone number is (571)270-1920. The examiner can normally be reached on Monday thru Friday, 9:00am to 6:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Jeffrey C. Pwu can be reached on (571) 272-6798. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Farhad Ali/
Examiner, Art Unit 2478

/Kenny S Lin/
Primary Examiner, Art Unit 2478